01

学习一门语言的路线图

（1）了解语言的背景,历史,特点,应用领域,现状.

（2）搭建一个开发环境,编写一个hello world.

（3）声明变量和常量.

（4）数据类型

（5）运算符

（6）逻辑结构

（7）通用的小程序

（8）函数和对象

（9）第三方的类库，插件，组件，框架...

（10）使用项目

1.js的运行环境

（1） 运行在浏览器端，使用浏览器引擎所自带的解释器。

（2）运行在服务器端，使用NodeJS解释器，基于谷歌的v8引擎

说明：js在浏览器端存在兼容性的问题，因为有众多浏览器品牌；而在服务器端不存在兼容性问题，因为服务器只有nodejs解释器

2.js的语法规范

（1）单行注释： //，多行注释： /\*注释内容\*/

（2）js中严格区分大小写

（3）每行结束后，可以不加分号，严格意思上讲需要加分号

3.变量

变量是一个用于存储信息的容器

（1）创建变量，使用关键字var （variable）

var a;

（2）使用“=”为变量赋值 var a = 'hello';

'=' 的作用把右边的值赋值给左边的变量

var b; //undefined

说明：变量只是声明了没有赋值返回undefined

变量可以一次性声明多个，多个之间用逗号隔开

var c=10,d;

(3)命名规范

1） 可以以字母，$, \_开头，后面可以出现数字，字母 ，\_，$。

2）连词之间使用下划线命名法，大驼峰命名法，小驼峰命名法（推荐使用）

3）不能使用关键字和保留字作为变量的名称

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| break | case | catch | continue | default |
| delete | do | else | false | finally |
| for | function | if | in | instanceof |
| new | null | return | switch | this |
| throw | true | try | typeof | var |
| void | while | with | undefined | ... |

练习：创建变量保存员工的姓名，性别，年龄，电话，邮箱。

**变量的预加载**

在代码执行的过程中，会将所有的变量声明提升到程序的最前端（var empname）,但不赋值。

4.常量

常量就是不能改变的值。

例如：圆周率，国家的面积

使用 const关键字来声明常量，习惯上，常量的名称全部使用大写，

5.数据类型

分为原始类型（值类型）和引用类型。

（1）原始类型

分为数值型，字符串类型，布尔型，undefined，null。

数值型又分为整数和浮点数

1)整数型

分为二进制，八进制（010）、十进制，十六进制（0XF）

2)浮点型

使用小数点记录的数据,1.2 3.14....

使用指数记录的数据,3.4E3

课后练习

声明一个常量保存圆周率，计算半径为4的圆的面积和周长

要求圆周率为常量，半径为变量保存数据

**02**

复习js

客户端->浏览器本身所自带的引擎（js解释器）

服务器端->nodejs所带的解释器，谷歌v8引擎

js变量

var a = 'hello world'; 以$, \_ , 字母开头，后面跟$, \_ , 字母

数值类型分为原始类型和引用类型

（1）数值型；整数和小数

整数在内存中占4个字节，小数占8个字节

1byte=8bit

整数分为16进制，8进制，10进制

（2）字符串类型

作用：常用于表示一些文本字符数据，例如：姓名，邮箱。。。

特点：使用引号（单引号和双引号）把数据所包含

查看汉字的Unicode编码 '北'.charCodeAt()

查看汉字Unicode的16进制形式

‘北'.charCodeAt().toString(16);

通过一个16进制的Unicode编码转成汉字

console.log('\u4e00');

汉字的Unicode编码范围4e00~9fa5

（3）（布尔型）

在程序中表示真或假的结果

取值 true/false

常用于一些表示是否为会员的结果

例如：是否为会员 var isMember = true;

（4）undefined 未定义型

在这种类型数据中，只有一个值undefined当声明了一个变量，此时变量的值就是undefined

（5）null

在js中，null表示一个不存在的对象，只有一个值null。

2.数据类型转换

（1）隐式（自动）转换

①数字+字符串：数字会被转成字符串

10+'kb' // '10kb'

②数字+布尔型：布尔型会被转成数字；true为1，false为0

2 + true = 3 2 + false =2

③字符串+布尔型：布尔型会被转成字符串

'hello'+true = hellotrue

④布尔型+布尔型：布尔型会被转成数字true为1，false为0

true+false=1

练习：检查一下程序的输出结果

（2）强制转换

①将任意的类型转成字符串类型

toString()

var a = 1;

a.toString() //1

②将任意的类型转换成整型

parseInt()

var num2 = '5.3ab';

parseInt(num2) //5

如果转换失败，返回NaN->Not a number

③将任意类型强制转换成浮点型

parseFloat()

var num3 = '5.3abc';

parseFloat(num3); //5.3

如果转换失败，返回NaN

④将任意类型转换成数值型

Number()

var num4 = '5.35';

Number(num4); //5.35

如果转换的数值中出现了非数字，结果为NaN

3. 两个浏览器端函数

prompt(); 弹出提示窗口

alert(); 弹出警示窗口

练习：弹出两次提示窗口，输入数字，打印出两个值相加的结果

**4.运算符和表达式**

**（1）算数运算符**：+ - \* / %取余 ++自增运算符，做加1运算 --自减运算符，做减1运算

console.log(num++);// //先打印出num结果，在加1

console.log(++num);// //先加1，在打印出num的结果

练习

var a = 3;

console.log(a++ + ++a);//8

**（2）关系运算符**（比较运算符）：< > ==(等于) >= <= != ===（全等） !==（不全等）

关系运算符的运算结果是布尔型的值（true/false）

==：只比两个值是否相等

===：既比较值，同时也比较类型是否相等

3>'10' //false

在关系运算符的两端，只要有一个是数值型，另一个也会隐式转换成数值型，在进行比较

'3'>'10' //true

字符串比较，比较的是Unicode的值

'3'>'10' 实际上比较的值是3和1 的Unicode的值

3=51 1=49

如果关系运算符的两端首个字符相同，继续往后比较第二个字符

'雷锋'>'雷峰' 锋=38155 峰=23792

'3a'>10 Number('3a') = NaN

注意：任何数值和NaN比较，结果都是false，包括本身

**（3）逻辑运算符**：或者 || 并且 && 非 !

||：关联的两个条件只要满足其中的一个，结果就为true

&&：关联的两个条件都满足，结果才是true，否者false。

！：取反

true && true ---->true

true && false ---->false

true || false ---->true

true || true ---->true

**逻辑短路**

短路逻辑&&，当第一个条件为false的时候，整体的表达式就是false，不需要再判断第二个条件。只有第一个条件为true的时候，才会继续判断第二个条件。

短路逻辑||，当第一个条件为true的时候，就不需要再去判断第二个条件，只有第一个条件为false的时候，才会继续判断第二条件。

（4）位运算符（了解）

在执行运算的时候，会把数字转换成二进制进行运算

按位与（&） 3 & 5 上下两位都是1结果才是1，否则0

按位或（|） 3 | 5 上下两位有一个是1，结果就是1

按位异或（^） 3^5 上下两位不同为1，相同为0

按位右移（>>） 15>>1=7 往右移变小

按位左移（<<） 15<<1=30 往左移变大

练习： 计算 7^10=13 5&8=0 10>>2=2

（5）赋值运算符（=）

扩展运算符 += -= \*= /= %=

练习：用户输入金额，如果输入的值享受8折优惠，输出的实际金额是多少？

（6）条件（三目）运算符

单目（一目/一元）运算符，只需要一个操作或者表达式 a++ a-- !true

双目（二元）运算符，需要两个操作或者表达式 + - \* / < > == != === !== <= >= || && & | ^ >> <<

+= -= \*= /= %=

三目（三元）运算符，需要三个操作或者表达式

条件表达式; ？ 表达式1 ： 表达式2

如果条件表达式为true 执行表达式1

如果条件表达式为false执行表达式2

练习：判断一个人是否为成年人，是 打印 成年人，否 打印 未成年人

**复习**

数据类型:原始类型(基本数据类型)和引用类型

数值型, 字符串类型,布尔型 undefined ,null

数据类型的转换:隐式转换 ，强制转换 .toString(); parseInt(); parseFloat(); Number();

运算符和表达式

算术运算符 关系运算符 逻辑运算符 按位运算符 赋值运算符 三目运算符

浏览器端函数

弹出警示窗口 alert()

弹出提示窗口 prompt()

**03**

**今日学习目标**

流程控制

循环语句

# 1.流程控制

程序 = 数据+算法

任何复杂程序的算法都可以通过“顺序执行”，“选择执行”，“循环执行”。

**（1）if语句**

选择执行某一些语句，

语句1

if(逻辑表达式){

语句2;

语句3;

....

}

语句4

执行流程：

①执行语句1

②执行逻辑表达是的值

如果值为true，则执行if中的语句2,3...

如果值为false，则不执行if中的语句...

③执行语句4

练习： 手动输入商品的价格以及商品的数量，计算产生商品的总价，

在if语句中的逻辑表达式，一些情况默认就是false 如：undefined, null ,0, NaN, '',

if 语句中如果大括号之间只有一行语句是可以省略大括号的

if(score>=60)

console.log('及格了')

**（2）if...else 语句**

语句1

if(逻辑表达式){

语句2;

}else{

语句3

}

语句4;

执行流程

①执行语句1

②判断if中的逻辑表达式的值

如果值为true，则执行语句2

如果值为false，则执行语句3

③执行语句4;

练习：手动输入商品的价格以及商品的数量，计算产生商品的总价，判断总价是否大于等于500，满足打八折，否则计算找零

**（3）if else 语句的嵌套**

用于判断多种情况

语句0;

if(逻辑表达式1){

语句1;

}else if(逻辑表达式2){

语句2;

}else if(逻辑表达式n){

语句n;

}else{

语句n+1; // 以上所有的逻辑表达式为false

}

①执行语句0

②执行逻辑表达式1;

若为逻辑表达式true，执行语句1

若逻辑表达式为false，则执逻辑表达式2

若逻辑表达式2为true，执行语句2

若逻辑表达式2为false，执行逻辑表达式n...

③以上所有的逻辑表达式为false，执行语句表达式n+1

# 2.switch...case 语句

是一种特殊的分支语句，可以根据一个表达式的不同值，来选择不同的程序入口来执行。

语句0;

switch(表达式){

case 1: //如果表达式的值1，选择进入这个入口执行程序

语句1;...

break; //终止，不会再往后执行

case 2: //若果表达式的值是2，选择进入这个入口执行程序

语句2; //

break;

default: //以上的值都不匹配，选择进入这个入口执行程序

语句n;

}

练习：

使用switch ...case来重新编写12星座特点

对比if-else嵌套和switch-case语句

相同点：两者都可以用分支语句

不同点：if-else可以判断相等和不等的情况，适用范围更广，

switch-case语句结构更加清晰，不适用于不等的情况，执行效率更高（速度快）

以后能使用switch-case来实现的效果尽量使用这个。

**3. 循环执行**

循环：就是一遍又一遍执行相同或者相似的代码。

循环节结构的两个要素：

（1）循环体：就是要重复相同或者相似的代码

（2）循环条件：重复的次数，或者执行的循环的条件

1）while循环 语法：

while(循环条件){

//循环体

}

课后练习：

使用switch...case模拟atm机

取款-->取款中

存款-->>请放入存款

转账

查询

退出

复习:

程序执行方式:顺序执行, 选择执行,循环执行

if 语句 if(表达式){}

if...else if(表达式){}else{}

if...else嵌套 if(表达式){}else if(){}

switch...case

switch(表达式){

case 1:

语句1

break; }

练习：while循环

猜数字游戏，产生一个数字，在弹出的提示框中输入一个数字，...

**04**

**今日目标**

do-while循环

for循环

循环嵌套

**2）do-while循环**

do{

//循环体

}while(循环条件);

①先是通过do执行循环体一次，不管循环条件是否为true

②判断循环条件是否为true。

③如果循环条件为true，执行循环体

④再次判断循环条件是否为true，直到为false才会终止循环

**while和do-while的区别**

①while语句，先判断，在执行

②do-while语句，先执行，在判断

③当初始值不满足条件的时候，while不会执行，do-while会执行一次

练习：先初始一个值，弹出提示框，如果输入值相等，弹出“”login success“”并且结束循环，否则继续弹出提示框

**3）for循环**

for(表达式1;表达式2;表达式3){

//循环体

}

表达式1为初始值

表达式2为循环条件

表达式3为循环的变化（增加，减少）

练习1：计算1~100的和

练习2：计算1-100之间所有奇数的和，

练习3：计算1-20之间所有偶数的乘积

练习4

**4）对比continue和break**

continue：跳出本次循环，继续下一次循环

break：跳出整个循环，后续不在执行循环。

练习1：计算1~100的和，当和大于4000的时候跳出循环，输出总和

练习2:计算1~100之间不能被3整除的数字

练习3：打印出2000！~2100年之间所有的闰年

**5）对比三种循环**

while常用于当...的时候，优先选择

do-while如果业务逻辑中含有直到....，优先使用

for 如果业务逻辑中要获取一个确切的循环

**6）for 循环的嵌套**